Chapter 6

FWAP-Mine – Frequency Weighted Access Pattern Mining

Weighted Sequential Pattern Mining is an approach that treats various items in a sequence with varying weights. Weighted method models the real life sequence database in a better manner and obviously takes over the ordinary sequential pattern mining. Weighted sequential pattern mining can be adapted to mine web access patterns more efficiently from web log data. This chapter describes the basics of weighted pattern mining and presents a new weighted access pattern mining algorithm, 'FWAP-Mine' to mine all weighted access patterns in a web log database. The new method uses frequency of user visit to give weights to web pages during the mining process. Through extensive experimental evaluation the algorithm is proved to be promising.

6.1 Introduction

Web servers trace and gather information about user interactions every time user interacts with the Web for some resources. So, the easiest way to find information about the users' navigation is to explore the Web server logs. Sequential pattern mining is an efficient technology used for extracting access patterns. Mining access pattern is the most time consuming and tedious job in web recommendation, web caching, web personalization etc. Two efficient and scalable methods for mining access patterns and maximal access patterns are proposed in Chapter 4 and Chapter 5.

But the main limitation of the traditional approach for mining frequent patterns and sequential pattern is that all items are treated uniformly. But, in real life examples, items have varying importance.

For Example, in conventional sequential pattern mining, a sequential pattern {(bread, milk) (diaper, beer)} can be easily discovered with support threshold because the support (frequency) of the sequential pattern is relatively high. Valuable (important) itemlists or sequences such as {(gold ring, silver necklace) (car, cell phone) and (computer, television)} cannot be mined with the previous sequential pattern mining approach because the items within the sequence are so expensive that the items have

low frequencies (supports). However, the items that are cheap and very common give low profits, and thus they are less important in terms of prices (weights) than low frequent but high profit items. In real business, marketing managers or trend analyzers would like to find the sequences with more emphasis on some particular (important) products (items) giving less emphasis on other products [Makris et al., 2007]. However, the sequential pattern mining approaches discussed in previous chapters cannot discover the important sequences with low supports. But in real life, items with low support become more important due to some features of the item itself.

For this reason, Weighted Pattern Mining algorithms have been suggested and it attaches varying weights to items according to their significance. Recently several proposals incorporating weight constraints into both Frequent Pattern Mining [Ahmed et al., 2008; Yun and Leggett, 2005; Yun, 2008, Yun, 2009] and Sequential Pattern Mining [Chen and Huang, 2003; Chen, Chiang and Ko, 2005; Chang, 2011] have been introduced.

Downward closure property is the main idea used in the pruning step of Frequent Pattern Mining. It says that a non frequent sequence can never lead to a frequent pattern and thus can be pruned from further explorations. But the downward closure property is usually broken when different weights are applied to the items. In Weighted Sequential Pattern Mining, the anti-monotone property cannot be directly used. The reason is, even if a sequential pattern is weighted infrequent, its super patterns may be weighted frequent because a sequential pattern with a low weight can attain high weight when another item with a higher weight is added to it. By introducing a global weight, the anti-monotone property can be maintained [Chen and Huang, 2003; Makris et al., 2007; Yun and Leggett, 2005].

Many of the previous work in this area [Chen and Huang, 2003; Forsat, Meybodii and Neiat, 2009; Tao, Farid and Murtagh, 2003; Yun and Leggett, 2005; Yun, 2008] assumed predefined weights for each item. But predefined weight is not much meaningful in web log analysis, as the importance of pages depends upon the user access itself. In this chapter, a new weighted access pattern mining method, FWAP-Mine, is proposed in which weights are assigned to each item based on the access sequence itself. Moreover, the earlier works involves two database scan in the generation of weighted sequential patterns. First scan is used to find out the frequent item set and the second one is for generating the set of access patterns using only the

frequent items. The proposed FWAP-Mine involves only one database scan. The non frequent items are not deleted from the WASD. Instead, they are elegantly skipped while the sequences are being processed. This enables the algorithm to easily get adapted to incremental mining. The method uses pattern growth based suffix building technique.

6.2 Sequential Pattern Mining

Sequential pattern means the pattern in which the order of elements in the input sequences is maintained while generating patterns. Sequential pattern mining is the mining of all patterns in an input sequence database that satisfy the minimum support criteria. In Section 3.2, a detailed discussion of *Sequential* Pattern Mining and related theory is given.

6.3 Weighted Sequential Pattern Mining

In conventional sequential pattern mining methods, all sequences in the database are treated with same importance. But in real life examples, sequences differ in their significances.

For this reason, weighted sequential pattern mining was introduced [Ahmed et al., 2008, Chen; Chiang and Ko, 2005; Chen and Huang, 2003]. Here, items within a sequence are given different weights according to their importance in the sequence database.

Item weight w(i) is defined as a value attached to an item to represent its significance. Let X=(x_1, x_2 …, x_n) be a set of distinct items and W be a set of non-negative real numbers. A pair (x, w) is called a weighted item where x Є X is an item and w Є W is the weight associated with x. A transaction is a set of weighted items, each of which may appear in multiple transactions with different weights.

Weight of an itemset I={x_1, x_2…, x_k} is derived from the weights of its enclosing items [Wang, Yang and Yu, 2000]. A simple way to calculate the average value of the item weights is,

$W(I) = \frac{\sum_{i=1}^{k} W(Xi)}{k}$, where k is the length of sequence (6.1)

Thus, the weight of the sequential pattern is the average value of the weights of items in a sequence. The weighted support, *wsup*, of a weighted sequential pattern is defined as the resultant value of multiplying the pattern's support with the weight of the pattern.

If *p* is a weighted pattern, the weighted support of *p*, wsup(*p*) is defined as

$$\mathrm{wsup(p)} = \mathrm{sup\ (p)} \times \mathrm{W(p)} \qquad (6.2)$$

A sequential pattern is called a weighted frequent sequential pattern if the weighted support of the sequential pattern is not less than a minimum threshold.

The main concern in weighted mining is the breaking down of the anti-monotone property when applying weights directly. Anti-monotone property is the crucial property applied in frequent pattern mining [Pei et al., 2000]. It says that an infrequent sequence can never lead to a frequent super sequence during the mining process. This property is used for pruning the infrequent sequences. But in weighted pattern mining, even if a sequential pattern is weighted infrequent, its super pattern may be weighted frequent because super patterns of the sequential pattern with a low weight can get a high weight after adding other items or item sets with higher weight [Chen and Huang, 2003; Makris et al., 2007; Srivastava, Bhosale and Sural, 2005].

6.4 Weighted Access Pattern Mining (WAPM)

Web access sequences in Web Access Sequence Database contain valuable information regarding the user behaviour and user interests. *Section 3.3* gives a detailed illustration of Web Access Pattern Mining.

In conventional Web Access Pattern Mining, order in which the pages are accessed is taken into consideration. But in practice, the frequency of visit to a web page, the time spent on each pages, current topic of interest etc are also important in pulling out the access patterns. Depending on the importance, weights can be attached to each web page. Weight attached to web pages allows generation of more appropriate access pattern. This helps in reducing the volume of access patterns generated and in turn reduces the space requirement.

Given an access sequence database, WASD and a support threshold, the problem of weighted access pattern mining is to find the complete set of all weighted access patterns whose weighted supports are not less than the support threshold.

6.5 Related Works

C F Ahmed *et.al* presented a tree structure $IWFPT_{WA}$ (Incremental Weighted Frequent Pattern Tree based on Weight Ascending order) and an algorithm $IWFP_{WA}$ for incremental and interactive Weighted Frequent Pattern mining [Ahmed et al., 2008]. Through the performance analysis, the authors showed that the proposed tree structure and corresponding mining algorithm were efficient for incremental and interactive weighted Frequent Pattern mining.

The authors of [Ahmed et al., 2008] claim that their method requires only one database scan. But this is due to the fact that their method assumes that the list of possible items and their weights are already given. In frequent pattern growth method having two database scan, the first scan is for finding out the elements and their support. Also the method sorts the elements in a transaction in the order of weights. But this is not possible in the case of web access sequences as the order of page access is very important.

Unil Yun proposed a weighted sequential pattern mining framework and developed WSpan algorithm based on the prefix projected sequential pattern growth approach [Yun, 2008]. The work suggested a new approach to detect more important sequential patterns. Weighted sequential pattern is defined and two pruning methods are suggested to detect more appropriate weighted sequential patterns. A weight range is given to the items according to the priority or importance. Performance analysis shows that WSpan is efficient and scalable.

Major challenge when making improvement in traditional Association Rule Mining by introducing weight is the invalidation of downward closure property. In 2003, F. Tao et al. proposed a set of new concepts to adapt weighting in to the mining process [Tao, Farid and Murtagh, 2003]. Among them the major proposal was the introduction of weighted support instead of simple support. The authors proved that original downward closure property can be replaced by weighted downward closure property. A new algorithm called WARM (Weighted Association Rule Mining) is developed based on the weighted support model. The authors proved the method to be both scalable and efficient in discovering significant relationships in weighted settings.

Srivastava et al. proposed a method for web caching using weighted association rule mining [Srivastava, Bhosale and Sural, 2005]. Their work shows that Weighted

Association Rule Mining (WAR) technique can be used to capture both users' habit and interest. Weights are assigned to all the URLs stored in a web log to distinguish recently accessed URLs from the old ones. Recent accesses are given more weights as they provide information regarding the current taste of a user. To achieve this, the log file is divided into different windows and each window is assigned a number known as window-index, starting from the last window. So, the window with lowest window-Index contains most recently accessed URLs, which are at the end of the log file. The method assigns all the URLs, which are in window 1 (lowest window-Index), a highest weight, h-Weight, where $0 < h\text{-}Weight < 1$. Through this technique the authors ensured that the recently accessed URLs get the highest weight. Weighted apriori method is used to generate the frequent URL set. The frequent URLs set generated by WAR contain mix of URLs capturing both habit and interest of the users. These frequent URLs can be used to speed up the web access by prefetching the URLs and present to the user whenever it is requested.

An efficient method for modelling user navigation history was proposed by C Makris et al. in [Makris et al., 2007]. The proposed system groups navigation sessions into clusters. Then each of the clusters is represented by a weighted sequence and using these sequences a generalized weighted suffix tree is constructed. This structure is used as a web page prediction/recommendation tool. Proposed estimation of user's navigational intention can be exploited either in an on-line recommendation system in a website or in a web-page cache system. The method has the advantage that it demands a constant amount of computational effort per one user's action and consumes a relatively small amount of extra memory.

R. Forsat et al. developed a web recommendation algorithm in [Forsat, Meybodii and Neiat, 2009]. The authors extend the traditional association rule problem by associating a weight with each item in a transaction to reflect the interest of each item within the transaction. The novel recommendation algorithm is based on the proposed weighted association rule. The method extract weighted association rules of each URL from the web log data and similarity between active user sessions is calculated upon the weighted rules instead of an exact match for finding the best rule. Recommendation engine will then find the most similar rules to the active user session with the highest weighted confidence by scoring each rule in terms of both its similarity to the active session and its weighted confidence.

In [Lee and Park, 2007], the authors proposed a method for mining weighted frequent patterns from web traversals. The authors adopt the approach of representing web structure as a graph. The structure of Web site is modelled as a graph in which the vertices represent web pages, and the edges represent hyperlinks between the pages. Furthermore, user navigations on the Web site is modelled as traversals on the graph. For mining the frequent patterns, weight is attached to each vertex of the graph. Importance of the vertex is reflected by such vertex weight. With this weight setting, they presented the mining algorithm which takes the weights into account in the measurement of support. The authors used support bound to prune candidates that have no possibility to become weighted frequent in the future.

6.6 The FWAP-Mine Method

In weighted Access Pattern Mining methods, there are two main components (i) weight assignment and (ii) pattern mining. Weight assignment module deals with assigning weights to items and item sets. Pattern mining module is for mining weighted access patterns from the WASD by applying the concept of weight and weighted support.

6.6.1 Weight Assignment

In conventional access pattern mining, only the order of access is considered. But the frequency of visit to a particular page of a user is a very important information as it shows the user's interest on that page. In the proposed Frequency Weighted Access Pattern mining (FWAP-Mine), frequency of user visit to a page is used to give weights to each item in an access sequence. The mining algorithm considers the weight of a sequence and uses it for finding out the support.

Weight of an access event $\boldsymbol{a}_i$ in an access sequence *S* in the Web Access Sequence Database (WASD) is defined as,

$$weight_S(a_i) = \frac{|\{a_i \mid a_i \subseteq S \wedge S \in WASD\}|}{|S|} \qquad (6.3)$$

The weight of an access sequence *S* in a Web Access Sequence Database is the average value of the weights of the sequence. Given an access sequence A = {$\boldsymbol{a_1}$, $\boldsymbol{a_2 \ldots a_m}$), Weight of the access sequence *A* is formally defined as follows.

$$\text{weight}(A) = \frac{\sum_{i=1}^{m} W(a_i)}{|S|} \qquad \text{where } \boldsymbol{a}_i \in S \text{ and } S \in WASD \qquad (6.4)$$

The weighted support of an access sequence is defined as the resultant value of multiplying the sequence's weight with global support (GlSup) of the sequence.

$$\text{wsup}(S) = \text{weight(s)} \times \text{GlSup}(S) \quad (6.5)$$

Global support of a sequence is the support of the sequence in the whole database.

$$\text{GlSup}(\text{S}_\text{i}) = \frac{|\{\text{S}|\ \text{S}_\text{i} \subseteq \text{S} \wedge \text{S} \in \text{WASD}\}|}{|\text{WASD}|} \quad (6.6)$$

A weighted access pattern is an access sequence that satisfies the predefined weighted support.

6.6.2 Mining of Weighted Access Patterns

The proposed method uses the pattern growth techniques which are proved to be better than apriori methods. Patterns are generated by suffix building using projection databases. Downward closure property is maintained in the method by bounding the support by global weight of each item.

WAP-Tree is a tree structure introduced in by J. Pai et al. to hold access sequences in a very compact form to enable access pattern mining [Pei et al., 2000]. An efficient Web Access Pattern Mining algorithm, FOL-Mine, was introduced in Chapter 4. It is based on the concept of pattern growth technique employed in WAP-Tree but proved to be more efficient than all previous WAP-tree based mining algorithms. FOL-Mine uses a special linked structure to hold access sequences for processing [Section 4.3.2.1 and 4.3.3].

The proposed FWAP-Mine method uses a modified form of the structure used in FOL-Mine to hold the access sequences. The nodes of the structure are modified to hold the generated weight information of each item. The proposed method needs only one database scan to load access sequences into the structure and to generate associated weight information.

FOL is the basic data structure [Section 4.3.2.3] used in FOL-Mine to hold the first occurrence information of items during the mining of patterns in the intermediate projected databases. FOL manages the suffix building very efficiently. The node structure of FOL is modified to process the weighted support of sequences.

The proposed method comprises of three algorithms. The first algorithm *Main*, given in Figure 6.4, is the main algorithm of the method. It reads in Web Access Sequences

(WAS) from WASD, assigns weight to each element and updates information regarding items and their frequencies. The second algorithm FWAP-Mine is the recursive algorithm used for generating the weighted frequent patterns of WASD. Detailed steps of FWAP-Mine are provided in Figure 6.6. The third algorithm GEN-FO, in Figure 6.8, is used for generating the list of first occurrences, LFO. Algorithm FWAP-Mine makes use of this procedure to effectively manage the projected databases.

6.6.2.1 Data Structures Used

Data structures play an important role in improving the efficiency of pattern growth methods. Main data structures used in the proposed method and their purpose are described below.

Item List (IL): Linked list containing the items and their frequencies present in the *WASD*. Figure 6.1 shows the node structure of *Item List*.

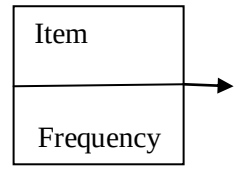

Figure 6.1: Node Structure of *Item List (IL)*

Current Item List (CIL): Linked list used for storing the items in the current access sequence under consideration and their frequencies present. Structure of CIL is same as *IL*.

List of First Occurrence (*LFO*): Linked list used for storing the first occurrences of a given item in the database. Each node contains the sequence-id (*sid*), and position of the occurrence (pos). Header node of the list holds the sum of weights of the element at each occurrence. Structure of LFO is given in Figure 6.2. If ***a*** is the given item and *D* is the input database, then *LFO* represents the ***a***-projections of ***a***, *Da*. Thus, *LFO* is used to manage the projection database very efficiently. During the generation of *LFO* infrequent elements are automatically skipped. So this arrangement of access sequence database does not require the deletion of infrequent elements unlike other earlier frequent pattern algorithms [Lee and Park, 2007; Pearson and Tang, 2008; Pei et al., 2000; Tang, Turkia and Gallivan, 2007; Yun and Leggett, 2005; Yun, 2009; Zhou, Hui and Fong, 2004]. So, the database structure is suitable for incremental mining too.

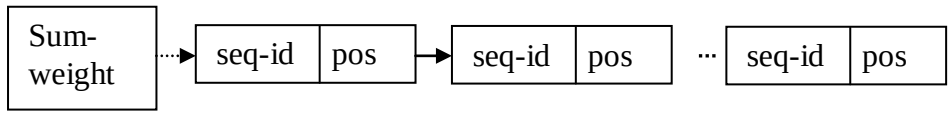

Figure 6.2: Structure of Linked List *LFO*

WASlist: Each web access sequence is stored in a linked list with node structure, *struct node{int item; int weight; next *node}.*

Headllist[]: The start address of each linked list containing item and its weight is registered in *Headlist[m*], where $m = \lfloor WASD \rfloor$.

Figure 6.3 depicts the structure for holding the Web Access Sequences.

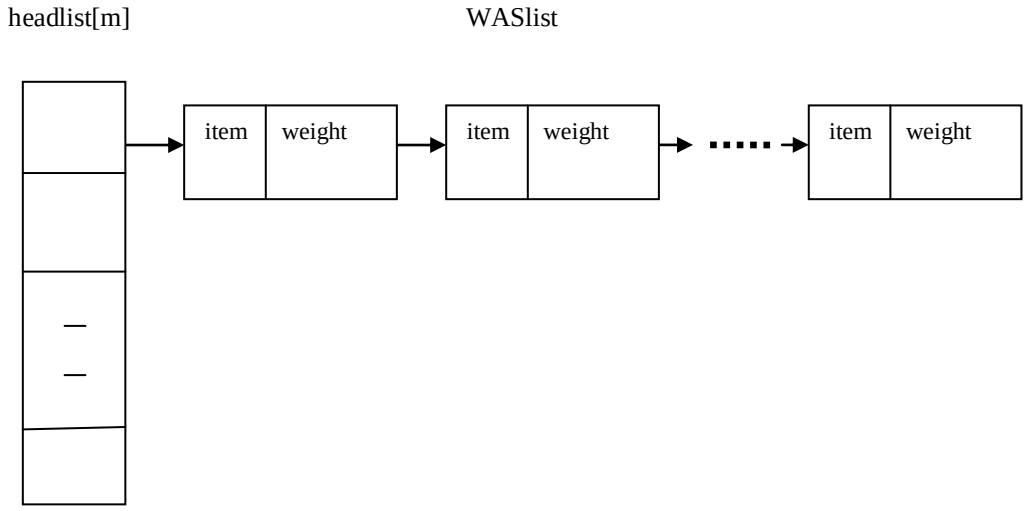

Figure 6.3: Structure of Headlist[ ] and Item List

6.6.2.2 Algorithm: Main

The algorithm reads access sequences one by one from WASD and stores it in the linked structure with weight of each item set to 0. During this scanning itself Current Item List, CIL, is loaded with items present in the current sequence and their frequencies. After finishing the scanning of one full access sequence, the list of all elements *IL* is updated using *CIL* to reflect the presence of elements and their frequencies in the current access sequence. Once this is over, weight of each element in *CIL* is calculated by dividing the frequency by the length of the sequence and weight in the current *WASlist* is updated.

Once the processing of access sequences is over, weighted frequent item set is generated using IL. If the frequency of an item is greater than the absolute support, it is considered

as a frequent event. Frequency of an item in the whole database is the global support GL-Support of the item. Then the recursive mining algorithm FWAP-Mine is called for generating the complete set of weighted access patterns. Algorithm of main is given in Figure 6.4 and the corresponding flow chart is provided in Figure 6.5.

Frequency weighted Access Pattern Mining Algorithm

Algorithm *Main*

Input:

An access sequence database, *WASD*

A support threshold

Output:

Set of weighted access patterns

Method:

1. While eof (*WASD*)

   Read in an access sequence $S = \boldsymbol{a}_1\boldsymbol{a}_2 \ldots \boldsymbol{a}_n$, assigning $weight(\boldsymbol{a}_i)=0$

   length =0

   For each element $\boldsymbol{a}_i$ in *S* do

   Increment *length*

   If $\boldsymbol{a}_i$ is not in C*IL*

   make an entry in *CIL* for $\boldsymbol{a}_i$ with $count(a_i)= 1$

   Else

   Increment the count of $\boldsymbol{a}_i$

   {end if}

   {*end for*}

   Update the list of items *IL* with the *CIL*

   For each $\boldsymbol{a}_{i..}$ update

   $weight\ (a_i) = count(a_i) / length$

   {*end while*}

2. Generate the set of weighted frequent items
3. Call FWAP-Mine
4. Return

Figure 6.4: Algorithm *Main*

6.6.2.3 Algorithm FWAP-Mine

FWAP-Mine is the recursive mining algorithm to generate weighted access patterns. It works as follows: - Consider the first element in the weighted frequent list. Algorithm Gen-FO is used to locate first occurrences in the current projected database. The weighted support returned along with the list of first occurrence is divided by the size of database, $|WASD|$, to maintain the downward closure property. If the weighted support wsup satisfies the support threshold, the weighted frequent element is stacked and FWAP-Mine is recursively called for further suffix building until support fails. Now the access pattern is generated and FWAP-Mine is again called with next weighted frequent element. This process is continued till no more patterns can be generated. Algorithm of FWAP-Mine is given in Figure 6.6 and Figure 6.7 shows the corresponding flowchart.

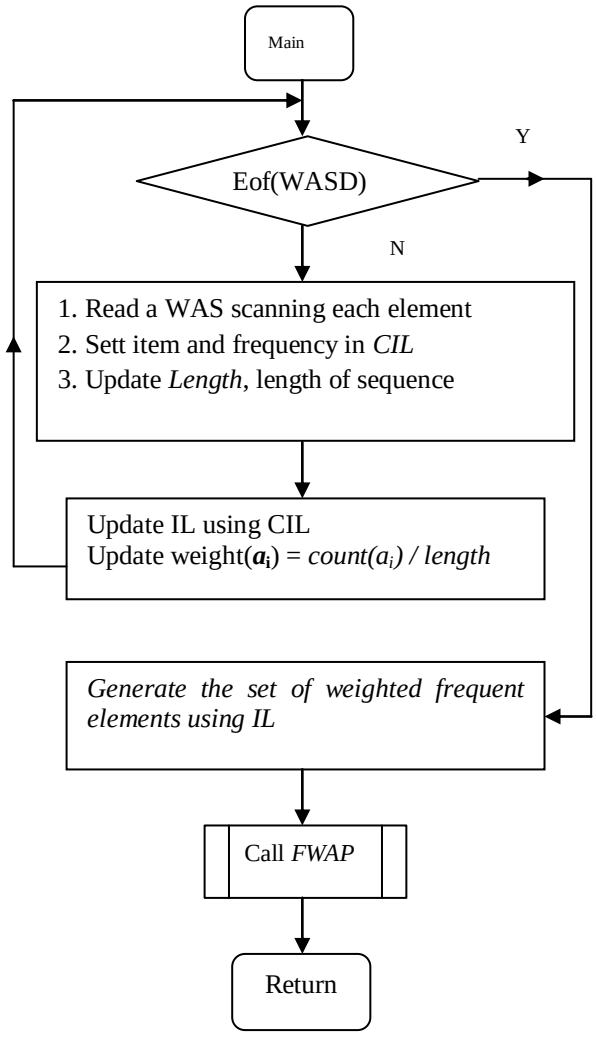

Figure 6.5: Flow chart of Algorithm *Main*

Recursive Weighted Access Pattern Mining Algorithm

Algorithm: FWAP-Mine

// E – The set of Patterns;

S – Stack of intermediate Frequent Patterns used for suffix building //

Parameters:

Current frequent pattern, *p*

List of fist occurrence, *L*

Absolute support, η

Method:

1. For each weighted frequent item, $\boldsymbol{a_i}$

i. Call *Gen-FO* to generate the first occurrences list, $L_{1,}$

ii. If the $WSup(a_i) > \eta$

Add $p.\boldsymbol{a_i}$ to *E*

Push $p.\boldsymbol{a_i}$ to *S*

p= $p.\boldsymbol{a_i}$

Call FWAP-Mine($p.,L1, \eta$)

{endif}

iii Delete the current *L*

{end for}

2. return

Figure 6.6: Algorithm *FWAP-Mine*

6.6.2.4 Algorithm GEN-FO

The algorithm GEN-FO works as follows: - In the initial call, input First occurrence List LFO is empty. So the algorithm uses the linked database itself to locate the first occurrence of the given item. In all the subsequent calls GEN-FO uses the input LFO, the one which is generated in the previous recursive call, to locate the first occurrences. Thus, LFO indicates the possible extensions in the projected database. This ensures the efficient suffix building. Weight of the element at each occurrence is added up. At the end, header of the first occurrence list is updated with the sum. New occurrence list L1 is returned to FWAP-Mine for further processing. Algorithm and flow chart of GEN-FO is given in Figure 6.8 and 6.9 respectively.

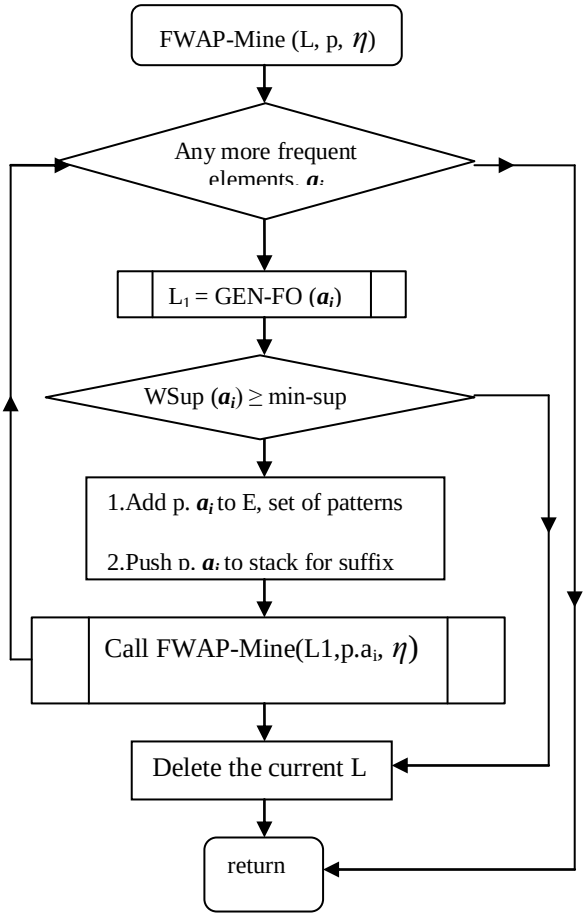

Figure 6.7: Flow Chart of Algorithm FWAP-Mine

Algorithm for generating List of First Occurrence, LFO

Algorithm GEN-FO

Parameters:

i. The list of first occurrence L (null in the initial call otherwise the list L generated in the previous call)

ii. current weighted frequent element, ***a***

Method:

1. Wsup = 0; Initialize L_1
2. If L is empty

   Locate first occurrences of ***a*** from the linked database.

   Generate L_1 with each node holding *seq-id* and *pos*

   Else

Locate the first occurrences of the element ***a*** in *Da* using L

Generate L_1 with each node holding seq-id and *pos*

{end if}

3. Add the weight of the item at each occurrence
4. Update the header of the list L_1 with total weight× GL-Support
5. return L_1

Figure 6.8: Algorithm *GEN-FO*

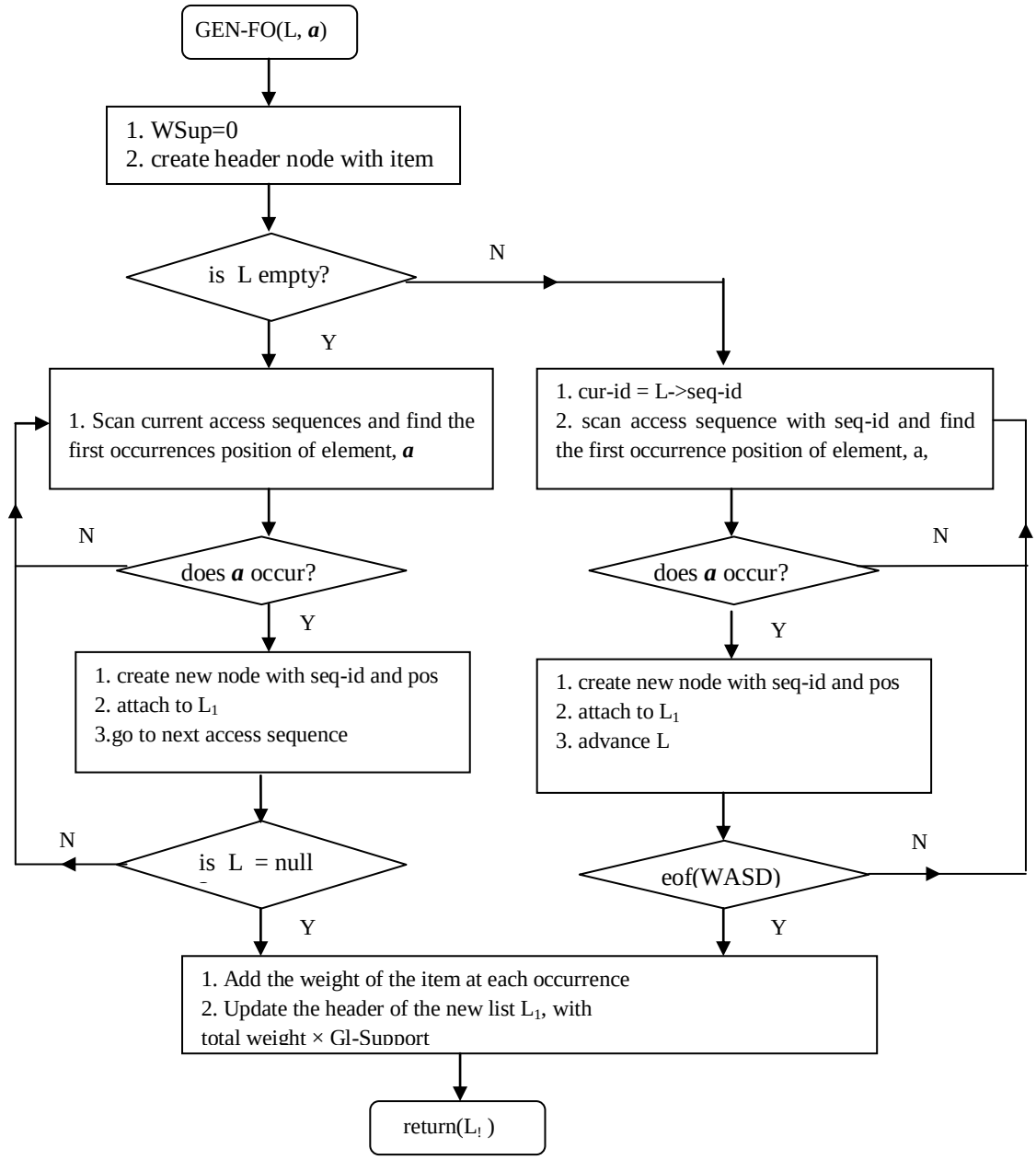

Figure 6.9: Flow chart of Algorithm *GEN-FO*

6.6.3 Illustration of Weighted Access Pattern Generation using FWAP-Mine Method

This section explains the generation of weighted access pattern using FWAP-Mine method with a small database of five sequences. Web Access Sequences in Table 4.1 is considered for illustration.

Step 1 of the main algorithm read access sequences one by one.

It start by reading the first sequence S_1 = ***abdac*** *and* load it in the structure with *weight 0. T*he sequence is loaded as:- ***a: 0, b: 0, d: 0, a: 0, c****: 0*; *the* corresponding *CIL* is in Figure 6.10 (i) and IL is shown in Figure 6.10 (ii).

a	*b*	*d*	*c*
2	*1*	*1*	*1*

a	*b*	*d*	*c*
2	*1*	*1*	*1*

Figure 6.10 (i): *CIL* **(ii):** *IL* after Reading the First Access Sequence

Length, the length of sequence is 5. Now, the weight of each element is updated as *Frequency/Length.*

So, the sequence becomes as: - ***a****: 0.4,* ***b****: 0.2,* ***d****: 0.2,* ***a****: 0.4,* ***c****: 0.2.*
It reads the second sequence S_2= ***eaebcac*** in to the structure *WASList* with weight sets to *0. CIL* is in Figure 6.11(i) *and IL* is in Figure 6.11(ii).

e	*a*	*b*	*c*
2	*2*	*1*	*2*

a	*b*	*d*	*c*	*e*
4	*2*	*1*	*3*	*2*

Figure 6.11(i): *CIL* **(ii):** *IL* after Reading the Second Access Sequence

Length of the sequence S_2=7. So, the weights are updated and sequence becomes as
S_2 = ***e****: 0.29,* ***a****: 0.29, .****e****: 0.29,* ***b****: 0.14, C: 0.29, a: 0.29, c: 0.29*

This process is continued with all sequences. The itemlist *IL* after reading all sequences is given in Figure 6.12.

a	b	d	c	e	f
8	5	1	6	3	3

Figure 6.12: *IL* after Reading All Access Sequences

Step 2 of the main algorithm finds the weighted frequent elements

sup = 0.75

absup = 0.75×4=3

Elements that have support >= 3 are selected as weighted frequent elements.

$\sum$ = { ***a, b, c***}

Step 3 of the algorithm call the recursive mining algorithm ***FWAP-Mine (Ɛ, null, η)***

Step1 of FWAP-Mine selects the first frequent item ***a*** for building the pattern

Step 1(i) of FWAP-Mine call GEN-FO (L-null, ***a***) to generate FOL, L_1

Step1 of GEN-FO set wsup = 0 and L1

Step 2 of GEN-FO find out the first occurrences of ***a*** from WASD as L is empty. This step generates L_1 as: - 1:1 —> 2:3 —> 3:2 —>4:2 and the sum of weights at each occurrence = total-weight=0.4+0.29+0.29+0.29=1.27.

Wsup=total-weight×global support=1.27×(8)

Thus L_1 is, **1.27:: 1:1—>2:3—>3:2—>4:2** and it is *returned to* FWAP-Mine.

Step 1(ii) of FWAP-Mine push ***a*** to stack if Wsup ≥ ***η***

p=p.***a***= ***Ɛ. a*** =***a***; add p to set of weighted frequent pattern

call FWAP-Mine = (p, L_1., *η)*

The recursive call FWAP-Mine again selects the first weighted frequent item *a* and call GEN-FO to find the first occurrence in the ***a*-projected database, Da,** as the input LFO L_1 is not empty search for first occurrence start from the positions in L_1.

L_2 is 1.27:: 1:4 —>2:6—>3:5—>4:4 and it is returned to FWAP-Mine

Step 1(ii) of FWAP-Mine push *a* to stack if *Wsup* ≥ ***η***

$p=p \,.\, \boldsymbol{a}= \boldsymbol{a} \,.\, \boldsymbol{a} =\boldsymbol{aa}$*;* add p to set of weighted frequent pattern now, the next recursive call FWAP-Mine(p, L_2., η*)* is initiated.

The recursive call FWAP-Mine again selects the first weighted frequent item ***a*** and call GEN-FO to find the first occurrence in the ***aa*-projected database, D*aa*.** As the input LFO L_2 is not empty search for first occurrence start from the positions in L_2.

L_3 is null and the current recursive call ends.

Step1 of the FWAP-Mine now consider the second weighted frequent element ***b*** *and call GEN-FO* to find the first occurrences in the projected database ***Dab***

This procedure is continued in **Step1** of FWAP-Mine with all frequent items. Process terminates when no more weighted pattern is to be generated.

6.7 Performance Evaluation

In this section we present the set of experiments that are performed for evaluating the performance of the proposed method. During the performance evaluation, focus was given to three different aspects, (i) Comparison of weighted and non weighted approach, both in terms of execution time and the number of patterns generated, (ii) Comparison of number of patterns generated and execution time requirement at various support threshold and (iii) Scalability of the method.

6.7.1 Test Environment and Datasets

For conducting the experiments three different datasets are used: two synthetic datasets, T25I10D10K and T10I4D100k and a real time dataset, *msnbc*. Description regarding the datasets are available in Section 4.4.1. These are standard datasets used in majority of sequential pattern mining studies [Chang, 2011, Pearson and Tang, 2008; Pei et al., 2005; Tang, Turkia and Gallivan, 2007; Yun and Leggett, 2005; Yun, 2008; Zhou, Hui and Fong, 2004].

All experiments were performed on a Intel Dual Core machine with 2GB RAM and running Microsoft Windows XP Professional version 2002. FWAP-Mine algorithm is implemented in Microsoft visual C++ 6.0.

6.7.2 Comparison of Weighted and Non Weighted Approach

Effectiveness of the Weighted Access Pattern Mining Method (FWAP-Mine) over the non-weighted approach is evaluated through both (i) comparing processing time and (ii) comparing the number of patterns generated.

The proposed method is compared with existing non weighted method to verify the advantage of the proposed method both in terms of memory and speed. For this purpose, non-weighted access pattern mining method *FOL-Mine* [Chapter 4] is selected.

6.7.2.1 Comparison of Processing Time

Experiments are done on different datasets to compare the execution time at different support values. Special stubs were inserted in the program to calculate the CPU time requirement for the execution of program.

Figure 6.13 shows the comparison of execution time of both methods graphically. The output of the experiment is provided in tabular form in Table 6.1. The result shows that the proposed FWAP-Mine method takes lesser execution time. Moreover as the support threshold decreases the execution time of the non weighted approach, FOL-Mine, increases at a higher rate.

Table 6.1: Execution Time & Number of Patterns Generated for FWAP-Mine and FOL-Mine for the Database T10I4D100K

Support	Execution Time(in sec)		Number of Patterns	
	FWAP-Mine	FOL-Mine	FWAP-Mine	FOL-Mine
0.004	49	135	26	2001
0.0035	58	155	39	2761
0.003	68	189	58	455 2
0.00 25	8 2	247	104	7703
0.00 2	10 2	314	15 2	13 255
0.0015	119	388	234	191 26

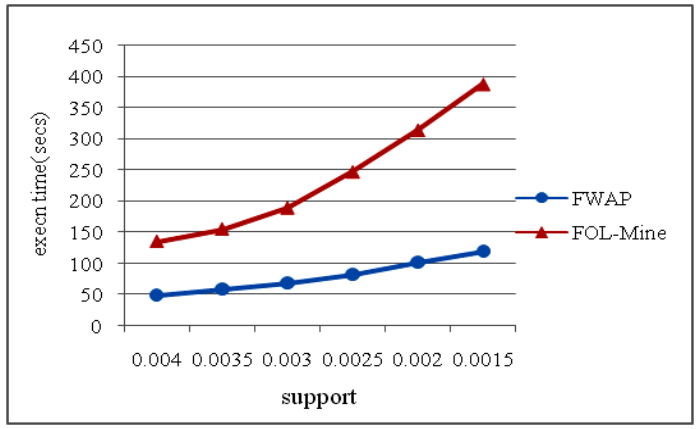

Figure 6.13: Execution Time Trend for FWAP-Mine and FOL-Mine for the Database T10I4D100K

6.7.2.2 Comparison of Number of Patterns Generated

Storing the huge volume of patterns generated is the primary concern in access pattern mining. To show how effectively the weighted pattern mining reduces the number of patterns, experiments are done with both weighted (FWAP-Mine) and non weighted (FOL-Mine) access pattern methods. Experiments are conducted on real data set *msnbc* and synthetic dataset T10I4D100K.

The result of experiment to compare the number of patterns by both FOL-Mine and the proposed FWAP-Mine on the datasets *msnbc* and T10I4D100K is provided in Figure 6.14 and Figure 6.15 respectively. The output of the experiment is available in tabular form in Table 6.1 and Table 6.2. From the graphs and the tables it is evident that the proposed method reduces the number of pattern generated by the introduction of new weight constraints.

Table 6.2: Comparison of Number of Patterns Generated for FWAP-Mine and FOL-Mine for the Database *msnbc*

Support	Number of Patterns	
	FWAP-Mine	FOL-Mine
0.004	282	2009
0.035	332	2483
0.003	410	30 24
0.00 25	533	3636
0.00 2	748	4199

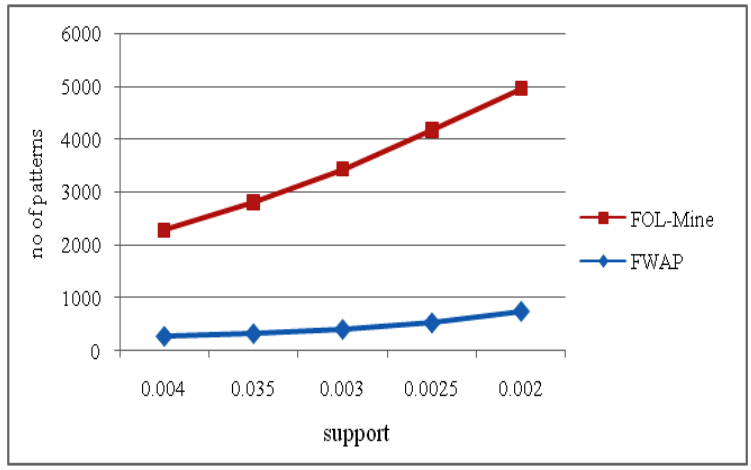

Figure 6.14: Comparison of Number of Patterns Generated for FWAP-Mine and FOL-Mine for the Database *msnbc*

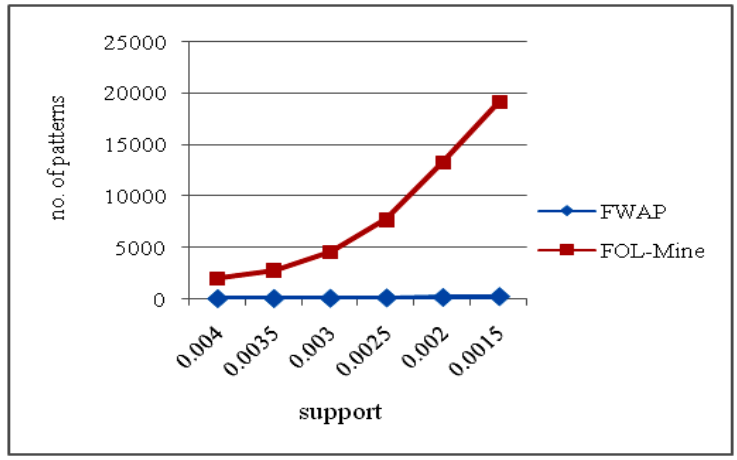

Figure 6.15: Comparison of Number of Patterns Generated for FWAP-Mine and FOL-Mine for the Database T10I4D100K

6.7.3 Comparison of Number of Patterns Generated and the Execution Time Requirement at Various Supports

In weighted access pattern mining we introduced the additional constraints of weight in to the mining process. This enables the method to pull out more meaningful patterns. Less significant patterns are neglected and this reduces the number of generated patterns. Managing the huge volume of pattern generated is an important concern in pattern mining.

The number of patterns generated in access pattern mining increases rapidly as the support threshold changes. This section evaluates the relation of execution time to the number of patterns generated at various thresholds.

Detailed illustration of the number of patterns generated at various support and the required execution time is provided.

The results of experiment using the dataset *msnbc* are given in Table 6.3 and Figure 6.16 and those for the data set T10I4D100K are provided in Table 6.1 and Figure 6.17. Number of patterns generated at various support for the dataset T25I10D10K is provided in Table 6.4 and Figure 6.18.

For all the three datasets, the proposed method shows a better performance. That is, by mining only the meaningful pattern the proposed method reduces the memory requirements.

Table 6.3: Number of Patterns Generated and Execution Time at Various Supports for the Database *msnbc*

Support	FWAP-Mine	
	Time	Number of Patterns
0.004	43	28 2
0.035	44	33 2
0.003	50	410
0.00 25	53	533
0.00 2	74	748

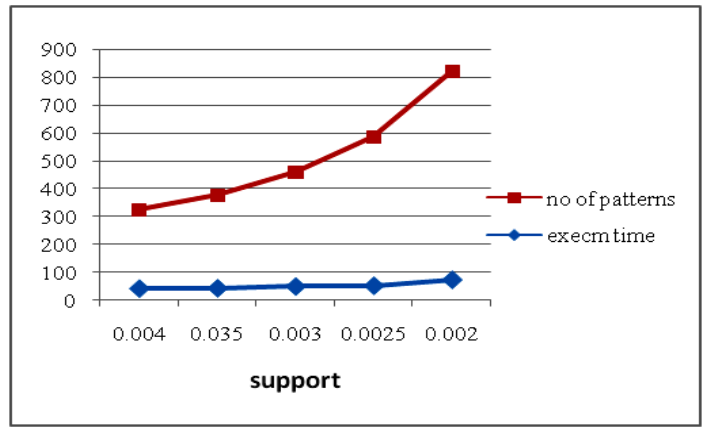

Figure 6.16: Number of Patterns Generated and Execution Time at Various Supports for the Database *msnbc*

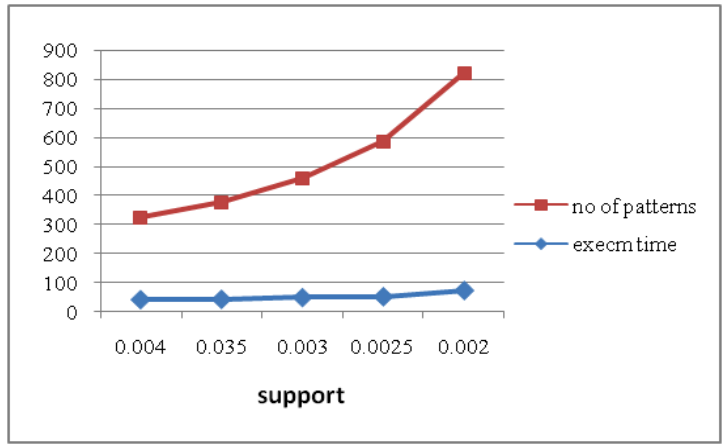

Figure 6.17: Number of Patterns Generated and Execution Time at Various Supports for the Database T10I4D100K

Table 6.4: Patterns Generated at Various Supports for the Database T25I10D10K

Support	Execution Time	Number of Patterns
0.004	10	22
0.0035	1 2	27
0.003	16	84
0.00 25	19	119
0.00 2	2 2	158
0.0015	31	223

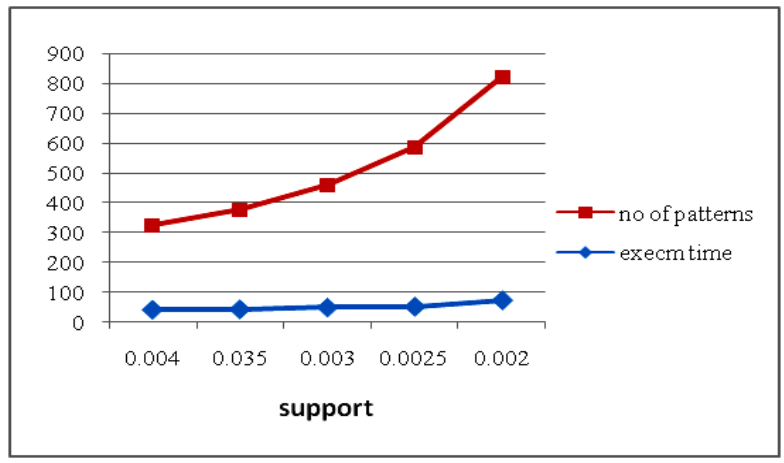

Figure 6.18: Patterns Generated at Various Supports for the Database T25I10D10K

6.7.4 Scalability Test

Scalability test tells how the algorithm performs when the size of input database increases. To test the scalability of the proposed algorithm experiments are done with both T25I10D10K and T10I4D100K. In the case of T25I10D10K dataset, scalability test are done by changing the database size from 2k to 10k. Table 6.5 and Figure 6.19 provide the results of these experiments.

For the database T10I4D100K, database size is changed from 20k to 100k to conduct the scalability test. Support threshold is maintained as 0.0015 in both cases. The results are shown in Table 6.6 and in Figure 6.20.

The results show that the performance of the proposed method is not affected by the size of the database and can efficiently work on larger databases. Moreover, FWAP has better scalability.

Table 6.5: Scaling up Experiments on T25I10D10K Database at Support Threshold 0.0015

data size(in K)	Execution Time(sec)	
	FWAP-Mine	FOL-Mine
2	3	5
4	9	10
6	16	17
8	24	23
10	31	31

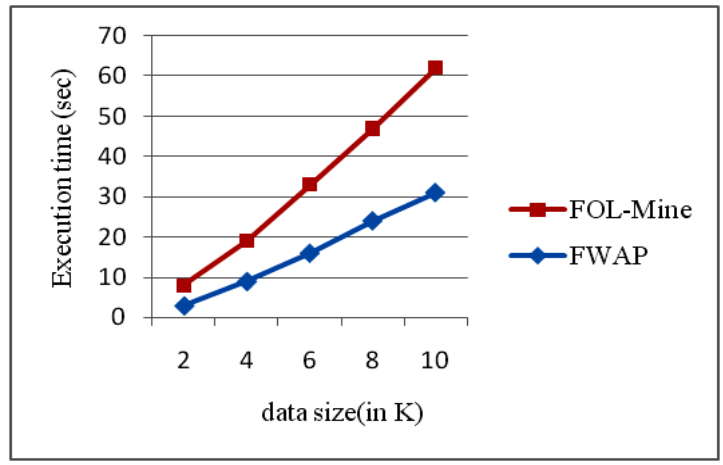

Figure 6.19: Scaling up Experiments on T25I10D10K Database at Support Threshold 0.0015

Table 6.6: Scaling up Experiments on T10I4D100K Database with Support Threshold 0.0015

Data size (in K)	Execution Time(in sec)	
	FWAP-Mine	FOL-Mine
20	16	38
40	33	97
60	57	161
80	83	2 29
100	117	29 2

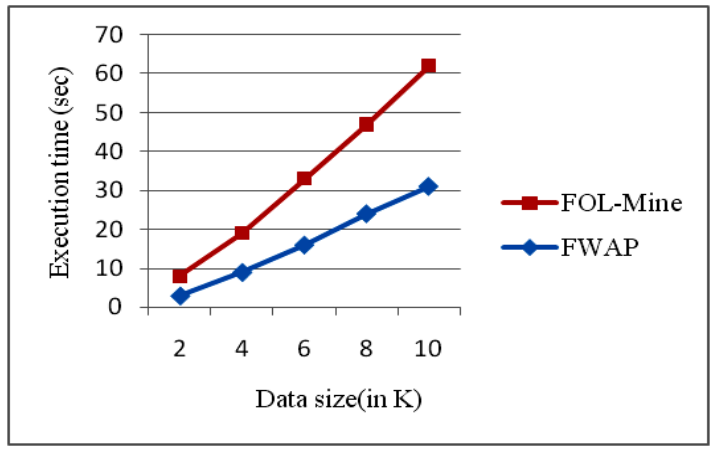

Figure 6.20: Scaling up Experiments on T10I4D100K Database with Support Threshold 0.0015

6.8 Summary

In this chapter a novel weighted access pattern mining algorithm, FWAP-Mine (Frequency Weighted Access Pattern mining), is introduced and described in detail. In weighted access pattern mining, not only the order of items in the sequence but also the relative importance of each item in the sequence is taken into consideration. Weights are attached to each item to signify their relative importance. In FWAP-Mine, frequency of user visit is used to give weights to each item in an access sequence. The proposed method is based on pattern growth suffix building heuristics. FWAP-Mine involves only one database scan and the data structure used may be easily adapted to incremental mining. Introduction of weight constraints lead to the pulling out of more meaningful frequent patterns.

Efficiency of the proposed method is evaluated in three different aspects – (i) Comparison of weighted and non weighted approach, both in terms of execution time and the number of patterns generated, (ii) Comparison of number of patterns generated and execution time requirement at various support threshold and (iii) Scalability of the method. Extensive evaluation of the algorithm based on the three aforesaid criteria proves better processing time and lesser memory requirement by the new method. Also, the new method is proved to be scalable and linear.